Interrupt Timing

Implementation

Fri, Aug 23, 2002

Task activity diagnostics have been in place for quite some time in the IRM/Linac systems, but a generic method for measuring interrupt timing has not. This note describes a scheme for such support that has been implemented in the IRM system software that is expected to be replicated in the Linac PowerPC system as well.

The key approach is to install in each interrupt routine whose timing is to be monitored a call to a function upon entry and a call to another function on exit. The elapsed time between is written into an Interrupt Timing Table entry. On a per-entry basis, options permit recording the time between interrupts or measuring the maximum execution time of the interrupt routine, or the minimum time between interrupts. In addition, one can select for each entry whether a LED is to be operated during interrupt execution, or whether a log is to be written of successive table records into a data stream. Besides the elapsed time, or any of its variations, the time of the interrupt relative to the present 15 Hz cycle is also recorded.

The means of time measurement is the microsecond counter in the IRM, or the Time Base register in the PowerPC. (The frequency of the latter is 16.667 MHz, but viewing the results on a memory display in hexadecimal, which is commonly done for many diagnostics, one can imagine the values seen as shifted right 4 bits, or one hexadecimal digit, to see values that approximate microseconds, with only a 4% over-estimate.) These sources are the most efficently available in the two environments.

A LED may or may not be assigned to each interrupt timing table entry. If it is, the hardware will be addressed. In the case of the PowerPC, there is significant time (about 1 μs) needed for each such access, for a total of 2 μs for each interrupt routine. But the PowerPC digital PMC module only delivers externally up to 4 LEDs out of 16 bits available for this purpose, using a multiplexing scheme. Even when a LED has been assigned to an interrupt routine, the support described here will not use it if that LED is not selected for external availability.

When viewing the Interrupt Timing Table, or ITT, a one-byte value serves as a reminder id to identify the use of that entry. The following id values have been assigned for use in the initial IRM-based implementation:

id value (hex)	*Interrupt activity*
`A0`	*Arcnet output
`A1`	*Arcnet input
`AD`	IRM 1KHz A/D
`Bn`	*Timer interrupt n
`C0`	Little console output
`C1`	Little console input
`C2`	Stepping motor
`CD`	Cycle delay
`CE`	Clock event
`CF`	Cycle 15 Hz
`D0`	Serial port output
`D1`	Serial port input
`E0`	*Ethernet output
`E1`	*Ethernet input

`15`	*1553 communications
`Bx`	*Dynamically assigned values

The interrupt routines marked with an asterisk are those not actually implemented yet. The IRMs do not use arcnet nor 1553, and an alternative diagnostic already exists for measuring ethernet interrupts. The dynamically-assigned values apply when a user program initializes an ITT entry specifying a zero value for the `id`.

The API for this facility includes the `ITTOpen` routine that is called when initializing an ITT entry for use.

```
FUNCTION ITTOpen(ntry, id, flagBit: Integer): ITTEPtr;
PROCEDURE ITTClose(ntryP: ITTEPtr);
```

The `ntry` parameter that specifies the desired ITT entry to be used, in the range 0–29. If the value –1 is used, an entry will be found that is not already assigned. The `id` value is taken from the list above, or it can be chosen arbitrarily. (Note that no alphanumeric ascii character codes are in the above list.) If the `id` value is zero, then a dynamically-assignable value will be chosen not being used by another entry. The system support implemented does not use this dynamic `id` option in its calls to `ITTOpen`. The `flagBit` value specifies an 8 bit value that uses the upper 4 bits for flags and the lower 4 bits for a LED bit number. The return value is a pointer to the applicable ITT entry. This return value can be used for calls to `ITTStart` and `ITTStop`, which are to be called at the beginning and end of each interrupt routine. The `ITTClose` routine is included for symmetry and merely frees up an ITT entry for another use, in case an interrupt routine is deactivated.

```
PROCEDURE ITTStart(ntryP: ITTEPtr);
PROCEDURE ITTStop(ntryP: ITTEPtr);
```

The ITT entries to be used are rather arbitrary, and could as well be assigned dynamically as the system is initialized. But it is useful to have the same entries be used across a set of similar nodes, so that one can easily capture the same entry information for comparison studies. Two ways are available to do this. One could, at boot time, copy the `id` values into the table entries prior to any calls to `ITTOpen`, then pick the option that specifies `ntry` = 0. The caller to `ITTOpen` would specify the appropriate `id` value, and the routine would find it in the table and use that entry. A second approach is to have the code know all the entries to be used in a standard way, via an include file. The latter approach was used in the first implementation. An include file knows all of the entries in use, all the `id` values to be used, all the LED assignments, and all of the initial flag settings. The calls to `ITTOpen` merely reference these named constants from the include file.

Two more routines are defined in the IRM package. One is called at 15 Hz to update certain fields in the ITT header. The other is called to initialize the log option. The entire ITT should be cleared to zero before any of the routines are called.

```
PROCEDURE ITTCycle;
PROCEDURE ITTInit;
```

The log option is based upon the data stream support in the system. This is used for many diagnostics as it facilitates being able to collect such diagnostics for analysis or display.

The format of the 32-byte ITT header is as follows:

Field	*Size*	*Meaning*
`ITTQPtr`	4	time within current accelerator cycle (`0x0C` event)
`ITTAct`	4	mask showing ITT entries active this cycle, so far
`ITTCycT`	4	base microsecond time at start of current cycle
`ITTActL`	4	value of ITTact at end of last cycle
`ILEDCpy`	2	copy of hardware interrupt LEDs register (PPC)
`TLEDCpy`	2	copy of hardware task LEDs register (PPC)
`ILEDMsk`	2	interrupt LEDs mux selection mask (PPC)
`TLEDMsk`	2	task LEDs mux selection mask (PPC)
`ITTDate`	8	time-of-day this cycle

The format of each of the 30 ITT fields is as follows:

Field	*Size*	*Meaning*
`id`	1	visual id as a reminder of the entry's use.
`flgBit`	1	flag bits and LED bit number
`count`	2	activity counter
`cycle`	4	time within current accelerator cycle (`0x0C` event)
`base`	4	base time
`elapsed`	4	elapsed time in raw counter units

All together, the required memory for this table is 512 bytes. In the IRM it is located at `0x800—0x9FF` in low memory. It can be at the same location in the PowerPC. (The PowerPC system maps requests for low memory into the area used for this purpose; vxWorks has its own designs on the use of low memory.)

Here is an example of the contents of the ITT:

```
FILE<0509>            08/23/02 1319
 0509:00000800   001F 2000 0000 0405 4F1F 20E5 0000 2D0F
     :00000810   0000 0000 0000 0000 0208 2313 1913 0600
     :00000820   CF00 C1A2 0000 0053 4F1F 20E7 0001 046E   0. 15 Hz
     :00000830   CD00 C1A1 0000 001C 4F1E B8E0 0000 9C67   1. 40 ms delay
     :00000840   CC86 C1A2 0000 003F 4F1F 20F6 0000 0011   2. CYCLE routine
     :00000850   CE8F 3141 0000 000D 4F1F 152E 0000 F8B5   3. Clock event
     :00000860   0000 0000 0000 0000 0000 0000 0000 0000   4.
     :00000870   0000 0000 0000 0000 0000 0000 0000 0000   5.
     :00000880   0000 0000 0000 0000 0000 0000 0000 0000   6.
     :00000890   0000 0000 0000 0000 0000 0000 0000 0000   7.
     :000008A0   D081 137D 0000 0009 4F1E 75D9 0000 5960   8. serial out
     :000008B0   D181 0000 0000 0000 0000 0000 0000 0000   9. serial in
     :000008C0   C084 0685 0000 0006 4F1F 2236 0000 0151  10. console out
     :000008D0   C184 0684 0000 0019 4F1E 683C 0000 4BC3  11. console in
     :000008E0   0000 0000 0000 0000 0000 0000 0000 0000  12.
     :000008F0   C282 8FCE 0000 0003 4F1F 11D9 0000 F560  13. motor step
```

This is the set of entries in use in the initial IRM implementation. The first two lines comprise the ITT header. The 4K-byte data stream queue used for the log option is at `0x1F2000`. The pattern `0x00002D0F` shows that all but one entry was active during the previous cycle.

The entries in use are listed on the right. The 15 Hz interrupt or the 40 ms delay interrupt may call the `CYCLE` routine. (In the absence of a 15 Hz interrupt, the occurrence of a second 40 ms interrupt causes `CYCLE` to be called.) In this case, the `CYCLE` routine was called by the 15 Hz interrupt routine, and one can see that its elapsed time of 63 μs is included in the 83 μs elapsed time of the 15 Hz interrupt. Since the 40 ms routine also does other functions, its execution reflects those jobs, which occur at 40 ms past the 15 Hz interrupt. In this example, the 15 Hz interrupt occurred at `0x1046E` μs after the start of the cycle; for this case, this refers to the start of the previous 15 Hz interrupt, and this value is 66670 μs, or very nearly one 15 Hz cycle. In contrast, one can see that the 40 ms interrupt occurred at `0x9C67` = 40039 μs after the start of the current cycle, which is very nearly 40 ms. A clock event interrupt occurred at `0xF8B5` = 63669 μs after the start of the previous cycle, which is not surprising, since the clock interrupt occurs 3 ms before the 15 Hz activity starts.

The serial output activity is shown, but there was no serial input interrupt detected. This is because the serial port was used for making this example listing via the Print Memory page application. Also, one can see the little console I/O that occurs on every cycle to sample the state of the little console hardware. The motor step interrupt normally occurs at 150 Hz. Its elapsed time is here measured at 3 μs. It will normally be short when there are no motor steps to be issued.

In all except two cases above, the flags are set so that the LED is operated. The flags are in bits 7–4 of the second byte of each entry. They have the following significance:

flagBit	*Mask*	*Meaning when set*
`flag7`	`0x80`	operate LED
`flag6`	`0x40`	elapsed time between successive interrupts, not within one interrupt
`flag5`	`0x20`	compute maximum/minimum elapsed time
`flag4`	`0x10`	log interrupts

Note that the `flag6` and `flag5` interact; when `flag6` is set, setting `flag5` measures the minimum time between interrupts, whereas when `flag6` is clear, setting `flag5` measures the maxmum time of interrupt execution.

When `flag4` is set, the log option is enabled, and copies of the ITT entry are logged into the `'ITTLOG  '` data stream, assuming that one was defined at boot time. Here is an example snapshot of this log:

```
:001F2180   CE9F 78DE 0000 000B 09BA 2328 0000 B6E7
:001F2190   CE9F 78DF 0000 000F 09BA 64FC 0000 F8BB
:001F21A0   CE9F 78E0 0000 0010 09BA F970 0000 88BC
:001F21B0   CE9F 78E1 0000 000B 09BB 2798 0000 B6E4
:001F21C0   CE9F 78E2 0000 000F 09BB 6970 0000 F8BC
:001F21D0   CE9F 78E3 0000 000F 09BB FDE4 0000 88BD
:001F21E0   CE9F 78E4 0000 000C 09BC 2C08 0000 B6E1
:001F21F0   CE9F 78E5 0000 000F 09BC 6DD9 0000 F8B2
:001F2200   CE9F 78E6 0000 000F 09BD 024D 0000 88BD
:001F2210   CE9F 78E7 0000 000A 09BD 3079 0000 B6E9
:001F2220   CE9F 78E8 0000 000F 09BD 724E 0000 F8BE
:001F2230   CE9F 78E9 0000 000F 09BE 06C3 0000 88BD
:001F2240   CE9F 78EA 0000 000B 09BE 34EA 0000 B6E4
:001F2250   CE9F 78EB 0000 000A 09BE 40A2 0000 C29C
:001F2260   CE9F 78EC 0000 000E 09BE 76C4 0000 F8BE
```

```
:001F2270   CE9F 78ED 0000 000F 09BF 0145 0000 7ECA
:001F2280   CE9F 78EE 0000 000B 09BF 0B37 0000 88BC
:001F2290   CE9F 78EF 0000 000F 09BF 395C 0000 B6E1
:001F22A0   CE9F 78F0 0000 000A 09BF 4514 0000 C299
:001F22B0   CE9F 78F1 0000 000F 09BF 7B34 0000 F8B9
:001F22C0   CE9F 78F2 0000 000E 09C0 05B5 0000 7ECA
:001F22D0   CE9F 78F3 0000 000B 09C0 0FA8 0000 88BD
:001F22E0   CE9F 78F4 0000 000B 09C0 3DCF 0000 B6E4
:001F22F0   CE9F 78F5 0000 0017 09C0 7FA9 0000 F8BE
:001F2300   CE9F 78F6 0000 000A 09C0 87DD 0001 00F2
:001F2310   CE9F 78F7 0000 000F 09C0 E156 0000 55F6
:001F2320   CE9F 78F8 0000 000B 09C1 047D 0000 791D
:001F2330   CE9F 78F9 0000 000B 09C1 0A2C 0000 7ECC
:001F2340   CE9F 78FA 0000 000A 09C1 0C4F 0000 80EF
:001F2350   CE9F 78FB 0000 000A 09C1 141F 0000 88BF
:001F2360   CE9F 78FC 0000 000A 09C1 423F 0000 B6DF
:001F2370   CE9F 78FD 0000 000E 09C1 841C 0000 F8BC
:001F2380   CE9F 78FE 0000 000F 09C2 1890 0000 88BE
:001F2390   CE9F 78FF 0000 000B 09C2 46AA 0000 B6D8
:001F23A0   CE9F 7900 0000 000E 09C2 8889 0000 F8B7
:001F23B0   CE9F 7901 0000 000F 09C3 1CFD 0000 88BD
:001F23C0   CE9F 7902 0000 000B 09C3 4B1A 0000 B6DA
:001F23D0   CE9F 7903 0000 000F 09C3 8CFC 0000 F8BC
:001F23E0   CE9F 7904 0000 000F 09C4 2170 0000 88BC
:001F23F0   CE9F 7905 0000 000C 09C4 4CF0 0000 B43C
:001F2400   CE9F 7906 0000 000A 09C4 4F8A 0000 B6D6
:001F2410   CE9F 7907 0000 000E 09C4 9170 0000 F8BC  0C,11
:001F2420   CE9F 7908 0000 0010 09C5 25E4 0000 88BD  18
:001F2430   CE9F 7909 0000 000B 09C5 53FB 0000 B6D4  0F
:001F2440   CE9F 790A 0000 000E 09C5 95DF 0000 F8B8
```

In this example of part of the log, only the clock event interrupt was selected for logging. The typical pattern of clock events is 4 that occur every 15 Hz cycle. (The 720 Hz event is ignored in typical IRM nodes.) Two occur at Booster reset event time, usually events `0x0C` and `0x11`, the `0x18` event occurs at about 38 ms past those two, and event `0x0F` occurs about about 50 ms past the first two. Sometimes more than 3 occur in the same 15 Hz cycle. For more detailed study of clock event activity, there is a page application designed for that purpose. But this diagnostic measures an time of about 15 μs for a clock event interrupt. The PowerPC is expected to be much faster.

Details

These modules were changed for the current IRM implementation:

`InzPSOS`	Soon after the call to `DSInit`, call `ITTInit` to set up log option.
`IntsFP`	Initializes and supports many of the system interrupts.
`ITTStart`	Package of new support routines for this feature.
`SerPort`	Serial port interrupt routines
`MotorInt`	Motor step interrupt
`RdAD`	IRM 1KHz A/D interrupt

The include file is `ITTDefs`.